Nota: Los ejercicios que tienen (S) son para una "Segunda vuelta" es decir conviene hacerlos una vez que ya se completó la guía haciendo los otros y ya se tiene mas madurez e intuición basica sobre los conceptos. Los que tienen (O) son opcionales por lo cual no se toman en los examenes.

Batallas entre paradigmas

En esta guia compararemos los tres paradigmas de computabilidad efectiva que hemos desarrollado anteriormente. Para esto probaremos que cada uno de dichos paradigmas "vence" al otro en el sentido que incluye por lo menos todas las funciones que incluye el otro en su modelizacion del concepto de funcion Σ-efectivamente computable. Por supuesto, esto dice que los tres son equivalentes.

Neumann vence a Godel

Usando macros podemos ahora probar que el paradigma imperativo de Neumann es por lo menos tan abarcativo como el funcional de Godel. Mas concretamente:

Theorem 1 *Si h es Σ-recursiva, entonces h es Σ-computable.*

Proof. Probaremos por induccion en k que

(*) Si $h \in \mathrm{R}_k^\Sigma$, entonces h es Σ-computable.

El caso $k = 0$ es dejado al lector. Supongamos (*) vale para k, veremos que vale para $k+1$. Sea $h \in \mathrm{R}_{k+1}^\Sigma - \mathrm{R}_k^\Sigma$. Hay varios casos

Caso 1. Supongamos $h = M(P)$, con $P : \omega \times \omega^n \times \Sigma^{*m} \to \omega$, un predicado perteneciente a R_k^Σ. Por hipotesis inductiva, P es Σ-computable y por lo tanto tenemos un macro

$$\big[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\overline{n+1}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}\big]$$

lo cual nos permite realizar el siguiente programa

$$\begin{array}{ll} \mathrm{L2} & \big[\text{IF } P(\mathrm{N}\overline{n+1}, \mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}) \text{ GOTO L1}\big] \\ & \mathrm{N}\overline{n+1} \leftarrow \mathrm{N}\overline{n+1} + 1 \\ & \text{GOTO L2} \\ \mathrm{L1} & \mathrm{N}1 \leftarrow \mathrm{N}\overline{n+1} \end{array}$$

Es facil chequear que este programa computa h.

Caso 2. Supongamos $h = R(f, \mathcal{G})$, con

$$\begin{array}{rcl} f & : & S_1 \times ... \times S_n \times L_1 \times ... \times L_m \to \Sigma^* \\ \mathcal{G}_a & : & S_1 \times ... \times S_n \times L_1 \times ... \times L_m \times \Sigma^* \times \Sigma^* \to \Sigma^*,\ a \in \Sigma \end{array}$$

elementos de R_k^{Σ}. Sea $\Sigma = \{a_1, ..., a_r\}$. Por hipotesis inductiva, las funciones f, $\mathcal{G}_a$, $a \in \Sigma$, son Σ-computables y por lo tanto podemos hacer el siguiente programa via el uso de macros

$$
\begin{array}{ll}
 & \left[\mathrm{P}\overline{m+3} \leftarrow f(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m})\right] \\
\mathrm{L}\overline{r+1} & \text{IF } \mathrm{P}\overline{m+1} \text{ BEGINS } a_1 \text{ GOTO L1} \\
 & \qquad \vdots \\
 & \text{IF } \mathrm{P}\overline{m+1} \text{ BEGINS } a_r \text{ GOTO L}\bar{r} \\
 & \text{GOTO L}\overline{r+2} \\
\mathrm{L1} & \mathrm{P}\overline{m+1} \leftarrow \ ^\frown\mathrm{P}\overline{m+1} \\
 & \left[\mathrm{P}\overline{m+3} \leftarrow \mathcal{G}_{a_1}(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}, \mathrm{P}\overline{m+2}, \mathrm{P}\overline{m+3})\right] \\
 & \mathrm{P}\overline{m+2} \leftarrow \mathrm{P}\overline{m+2}.a_1 \\
 & \text{GOTO L}\overline{r+1} \\
 & \qquad \vdots \\
\mathrm{L}\bar{r} & \mathrm{P}\overline{m+1} \leftarrow \ ^\frown\mathrm{P}\overline{m+1} \\
 & \left[\mathrm{P}\overline{m+3} \leftarrow \mathcal{G}_{a_r}(\mathrm{N}1, ..., \mathrm{N}\bar{n}, \mathrm{P}1, ..., \mathrm{P}\bar{m}, \mathrm{P}\overline{m+2}, \mathrm{P}\overline{m+3})\right] \\
 & \mathrm{P}\overline{m+2} \leftarrow \mathrm{P}\overline{m+2}.a_r \\
 & \text{GOTO L}\overline{r+1} \\
\mathrm{L}\overline{r+2} & \mathrm{P}1 \leftarrow \mathrm{P}\overline{m+3}
\end{array}
$$

Es facil chequear que este programa computa h.

El resto de los casos son dejados al lector. ■

Corollary 2 *Si*

$$
\begin{aligned}
f &: D_f \subseteq \omega^n \times \Sigma^{*m} \to \omega \\
g &: D_g \subseteq \omega^n \times \Sigma^{*m} \to \Sigma^* \\
P &: D_P \subseteq \omega^n \times \Sigma^{*m} \to \{0, 1\}
\end{aligned}
$$

son Σ-recursivas, entonces hay macros

$$
\begin{array}{l}
\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right] \\
\left[\mathrm{W}\overline{m+1} \leftarrow g(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right] \\
[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]
\end{array}
$$

Ejercicio 1: Pruebe el corolario anterior y justifique la siguiente aceveracion: "Hay macros

$$
\begin{array}{l}
\left[\mathrm{V}\overline{n+1} \leftarrow f(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right] \\
\left[\mathrm{W}\overline{m+1} \leftarrow g(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m})\right] \\
[\text{IF } P(\mathrm{V}1, ..., \mathrm{V}\bar{n}, \mathrm{W}1, ..., \mathrm{W}\bar{m}) \text{ GOTO A1}]
\end{array}
$$

para todas las funciones Σ-mixtas y predicados Σ-mixtos que hemos trabajado hasta el momento en la materia".

Como veremos, lo expresado en el ejercicio anterior transforma al lenguaje $\mathcal{S}^{\Sigma}$ en un potente y relativamente comodo lenguaje de programacion, el cual usaremos como herramienta para obtener interesantes e importantes resultados. Por ejemplo a continuacion usaremos la existencia de los macros [IF V1 es par GOTO A1] y [V2 $\leftarrow \lfloor$V1/2$\rfloor$] para probar el siguiente resultado cuya prueba esta inspirada en su analoga del paradigma de computabilidad efectiva (dejada como ejercicio en la Guia 3).

Lemma 3 *Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-enumerables. Entonces $S_1 \cup S_2$ es Σ-enumerable.*

Proof. Podemos suponer que ni S_1 ni S_2 son vacios ya que de lo contrario los resultados son triviales. Ademas supondremos que $n = 2$ y $m = 1$.

La idea de la prueba es la misma que la que usamos para probar que la union de conjuntos Σ-efectivamente enumerables es Σ-efectivamente enumerable. Daremos usando macros un programa que enumera a $S_1 \cup S_2$ y luego aplicaremos la proposicion del final de la Guia 7. Por hipotesis hay funciones $F : \omega \to \omega \times \omega \times \Sigma^*$ y $G : \omega \to \omega \times \omega \times \Sigma^*$ tales que $F_{(1)}$, $F_{(2)}$, $F_{(3)}$, $G_{(1)}$, $G_{(2)}$ y $G_{(3)}$ son Σ-computables, $\mathrm{Im}(F) = S_1$ y $\mathrm{Im}(G) = S_2$. O sea que hay macros

$$\begin{gathered}
\big[\mathrm{V2} \leftarrow F_{(1)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow F_{(2)}(\mathrm{V1})\big] \\
\big[\mathrm{W1} \leftarrow F_{(3)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow G_{(1)}(\mathrm{V1})\big] \\
\big[\mathrm{V2} \leftarrow G_{(2)}(\mathrm{V1})\big] \\
\big[\mathrm{W1} \leftarrow G_{(3)}(\mathrm{V1})\big]
\end{gathered}$$

Ya que el predicado $Par = \lambda x[x$ es par] es Σ-p.r., el Corolario 2 nos dice que hay un macro:

$$[\mathrm{IF}\ Par(\mathrm{V1})\ \mathrm{GOTO\ A1}]$$

el cual escribiremos de la siguiente manera mas intuitiva

$$[\mathrm{IF\ V1\ es\ par\ GOTO\ A1}]$$

Ya que el predicado $D = \lambda x[\lfloor x/2 \rfloor]$ es Σ-p.r., el Corolario 2 nos dice que hay un macro:

$$[\mathrm{V2} \leftarrow D(\mathrm{V1})]$$

el cual escribiremos de la siguiente manera mas intuitiva

$$[\mathrm{V2} \leftarrow \lfloor \mathrm{V1}/2 \rfloor]$$

Sea $\mathcal{P}$ el siguiente programa:

$$
\begin{array}{ll}
 & [\text{IF N1 es par GOTO L1} \\
 & \text{N1} \leftarrow \text{N1} \dot{-} 1 \\
 & [\text{N1} \leftarrow \lfloor \text{N1}/2 \rfloor] \\
 & \left[\text{N1} \leftarrow G_{(1)}(\text{N1})\right] \\
 & \left[\text{N2} \leftarrow G_{(2)}(\text{N1})\right] \\
 & \left[\text{P1} \leftarrow G_{(3)}(\text{N1})\right] \\
 & \text{GOTO L2} \\
\text{L1} & [\text{N1} \leftarrow \lfloor \text{N1}/2 \rfloor] \\
 & \left[\text{N1} \leftarrow F_{(1)}(\text{N1})\right] \\
 & \left[\text{N2} \leftarrow F_{(2)}(\text{N1})\right] \\
 & \left[\text{P1} \leftarrow F_{(3)}(\text{N1})\right] \\
\text{L2} & \text{SKIP}
\end{array}
$$

Es facil ver que $\mathcal{P}$ cumple a y b de (2) de la proposicion del final de la Guia 7 por lo cual $S_1 \cup S_2$ es Σ-enumerable. ■

Ejercicio 2: Pruebe que $\omega \times \omega \times \Sigma^*$ es Σ-enumerable (Hint: usando macros (asociados a las "bajadas" y a $*^{\leq}$) haga un programa que enumere a $\omega \times \omega \times \Sigma^*$, es decir que cumpla a y b de (2) de la proposicion del final de la Guia 7)

Ejercicio 3: Supongamos $S_1, S_2 \subseteq \omega^n \times \Sigma^{*m}$ son conjuntos Σ-enumerables. Entonces $S_1 \cap S_2$ es Σ-enumerable. (Hacer el caso $n = 2$, $m = 1$ e inspirese en el paradigma efectivo)

En la Guia 3 probamos que si $S \subseteq \omega^n \times \Sigma^{*m}$ es Σ-efectivamente computable entonces S es Σ-efectivamente enumerable. Via el uso de macros adecuados podemos copiar la idea de la prueba de dicho resultado para probar el siguiente

Lemma 4 *Sea $S \subseteq \omega^n \times \Sigma^{*m}$. Si S es Σ-computable, entonces S es Σ-enumerable*

Ejercicio 4: Pruebe el lema anterior (haga el caso $n = 2$, $m = 1$)

Ejercicio 5: Si $S \subseteq \Sigma^*$ es Σ-enumerable, entonces

$$T = \{\alpha \in \Sigma^* : \text{existe } \beta \in S \text{ tal que } \alpha \text{ es subpalabra de } \beta\}$$

también es Σ-enumerable.

Godel vence a Neumann

Primero definiremos tres funciones las cuales contienen toda la informacion acerca del funcionamiento del lenguaje $\mathcal{S}^{\Sigma}$. Sean $n, m \in \omega$, fijos. Definamos

$$\begin{aligned} i^{n,m} &: \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega \\ E_{\#}^{n,m} &: \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega^{[\mathbf{N}]} \\ E_{*}^{n,m} &: \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \Sigma^{*[\mathbf{N}]} \end{aligned}$$

de la siguiente manera

$$(i^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(0,\vec{x},\vec{\alpha},\mathcal{P})) = (1,(x_1,...,x_n,0,...),(\alpha_1,...,\alpha_m,\varepsilon,...))$$
$$(i^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t+1,\vec{x},\vec{\alpha},\mathcal{P})) = S_{\mathcal{P}}(i^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t,\vec{x}$$

Notese que

$$(i^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}))$$

es la descripcion instantanea que se obtiene luego de correr $\mathcal{P}$ una cantidad t de pasos partiendo del estado

$$((x_1,...,x_n,0,...),(\alpha_1,...,\alpha_m,\varepsilon,...))$$

Es importante notar que si bien $i^{n,m}$ es una funcion $(\Sigma \cup \Sigma_p)$-mixta, ni $E_{\#}^{n,m}$ ni $E_{*}^{n,m}$ lo son.

Definamos para cada $j \in \mathbf{N}$, funciones

$$\begin{aligned} E_{\#j}^{n,m} &: \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \omega \\ E_{*j}^{n,m} &: \omega \times \omega^n \times \Sigma^{*m} \times \mathrm{Pro}^{\Sigma} \to \Sigma^{*} \end{aligned}$$

de la siguiente manera

$$\begin{aligned} E_{\#j}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) &= j\text{-esima coordenada de } E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) \\ E_{*j}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) &= j\text{-esima coordenada de } E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) \end{aligned}$$

(es claro que estas funciones son $(\Sigma \cup \Sigma_p)$-mixtas). Notese que

$$\begin{aligned} E_{\#}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) &= (E_{\#1}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{\#2}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}),...) \\ E_{*}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}) &= (E_{*1}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}), E_{*2}^{n,m}(t,\vec{x},\vec{\alpha},\mathcal{P}),...) \end{aligned}$$

Aceptaremos sin prueba la siguiente proposicion.

Proposition 5 *Sean $n, m \geq 0$. Las funciones $i^{n,m}$, $E_{\#j}^{n,m}$, $E_{*j}^{n,m}$, $j = 1, 2, ...$, son $(\Sigma \cup \Sigma_p)$-p.r.*

Las funciones $Halt^{n,m}$ y $T^{n,m}$ Dados $n, m \in \omega$, definamos:

$$Halt^{n,m} = \lambda t \vec{x} \vec{\alpha} \mathcal{P} \left[i^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}) = n(\mathcal{P}) + 1 \right]$$

Notese que $D_{Halt^{n,m}} = \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma}$ (ojo que aqui la notacion lambda es respecto del alfabeto $\Sigma \cup \Sigma_p$). Ademas notese que usamos la variable $\mathcal{P}$ en la notacion lambda por un tema de comodidad psicologica dado que $i^{n,m}$ esta definida solo cuando la ultima coordenada es un programa pero podriamos haber escrito $\lambda t \vec{x} \vec{\alpha} \alpha \left[i^{n,m}(t, \vec{x}, \vec{\alpha}, \alpha) = n(\alpha) + 1 \right]$ y sigue siendo la misma funcion.

Cabe destacar que $Halt^{n,m}$ tiene una descripcion muy intuitiva, ya que dado $(t, \vec{x}, \vec{\alpha}, \mathcal{P}) \in \omega \times \omega^n \times \Sigma^{*m} \times \text{Pro}^{\Sigma}$, tenemos que $Halt^{n,m}(t, \vec{x}, \vec{\alpha}, \mathcal{P}) = 1$ si y solo si el programa $\mathcal{P}$ se detiene luego de t pasos partiendo desde el estado $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$.

Aceptaremos sin demostracion el siguiente lema (ver el apunte por una prueba).

Lemma 6 *(a)* Pro^{Σ} *es un conjunto* $(\Sigma \cup \Sigma_p)$*-p.r.*

(b) $\lambda \mathcal{P} \left[n(\mathcal{P}) \right]$ *y* $\lambda i \mathcal{P} \left[I_i^{\mathcal{P}} \right]$ *son funciones* $(\Sigma \cup \Sigma_p)$*-p.r..*

Ahora podemos probar el siguiente importante resultado.

Lemma 7 $Halt^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-p.r.*

Proof. Notar que $Halt^{n,m} = \lambda xy[x = y] \circ \left[i^{n,m}, \lambda \mathcal{P}[n(\mathcal{P}) + 1] \circ p^{1+n,m+1}_{1+n+m+1} \right]$. ■

Ahora definamos $T^{n,m} = M(Halt^{n,m})$. Notese que

$$D_{T^{n,m}} = \{(\vec{x}, \vec{\alpha}, \mathcal{P}) : \mathcal{P} \text{ se detiene partiendo de } \|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|\}$$

y para $(\vec{x}, \vec{\alpha}, \mathcal{P}) \in D_{T^{n,m}}$ tenemos que $T^{n,m}(\vec{x}, \vec{\alpha}, \mathcal{P}) =$ cantidad de pasos necesarios para que $\mathcal{P}$ se detenga partiendo de $\|x_1, ..., x_n, \alpha_1, ..., \alpha_m\|$. En algun sentido, la funcion $T^{n,m}$ mide el "tiempo" que tarda en detenerse $\mathcal{P}$ y de ahi su nombre

Proposition 8 $T^{n,m}$ *es* $(\Sigma \cup \Sigma_p)$*-recursiva*

Proof. Es directo del lema de minimizacion ya que $Halt^{n,m}$ es $(\Sigma \cup \Sigma_p)$-p.r. ■

Ahora nos sera facil probar que el paradigma de Godel es por lo menos tan abarcativo como el imperativo de Von Neumann. Mas concretamente:

Theorem 9 *Si* $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ *es* Σ*-computable, entonces* f *es* Σ*-recursiva.*

Proof. Haremos el caso $O = \Sigma^*$. Sea $\mathcal{P}_0$ un programa que compute a f. Primero veremos que f es $(\Sigma \cup \Sigma_p)$-recursiva. Note que

$$f = E^{n,m}_{*1} \circ \left[T^{n,m} \circ \left[p^{n,m}_1, ..., p^{n,m}_{n+m}, C^{n,m}_{\mathcal{P}_0}\right], p^{n,m}_1, ..., p^{n,m}_{n+m}, C^{n,m}_{\mathcal{P}_0}\right]$$

donde cabe destacar que $p^{n,m}_1, ..., p^{n,m}_{n+m}$ son las proyecciones respecto del alfabeto $\Sigma \cup \Sigma_p$, es decir que tienen dominio $\omega^n \times (\Sigma \cup \Sigma_p)^{*m}$. Esto nos dice que f es $(\Sigma \cup \Sigma_p)$-recursiva. O sea que el Teorema de Independencia del Alfabeto nos dice que f es Σ-recursiva. ■

Aceptaremos sin prueba la siguiente proposicion.

Proposition 10 *La funcion $T^{n,m}$ no es $(\Sigma \cup \Sigma_p)$-p.r.*

Corollary 11 *La minimizacion de un predicado Σ-p.r. no necesariamente es Σ-p.r.*

Proof. Por definicion $T^{n,m} = M(Halt^{n,m})$. ■

Uso de macros asociados a las funciones $Halt^{n,m}$, $E^{n,m}_{\#}$ y $E^{n,m}_{*}$

Aqui veremos, con ejemplos, como ciertos macros nos permitiran dentro de un programa hablar acerca del funcionamiento de otro programa. Esto junto con el hecho que cada funcion Σ-recursiva y cada predicado Σ-recursivo tienen su macro asociado (Corolario 2), sera muy util a la hora del diseño de programas y nos permitira simular dentro del paradigma imperativo muchas ideas usadas para el diseño de procedimientos efectivos. En este sentido la convinacion de los dos paradigmas (recursivo e imperativo) nos permite fortaleser notablemente al paradigma imperativo en su roll modelizador (o simulador) de los procedimientos efectivos. Esto es importante ya que el paradigma mas comodo, amplio e intuitivo, a la hora de decidir si algo es o no computable, es sin duda el filosofico o efectivo.

Sea $\Sigma = \{@, !\}$ y sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$ tal que $0 \in \text{Dom}\Psi^{1,0,\#}_{\mathcal{P}_0}$ y $\Psi^{1,0,\#}_{\mathcal{P}_0}(0) = 2$. Probaremos que

$$S = \{x \in \text{Dom}\Psi^{1,0,\#}_{\mathcal{P}_0} : \Psi^{1,0,\#}_{\mathcal{P}_0}(x) \neq 0\}$$

es Σ-enumerable. Notese que $0 \in S$. Por definicion de conjunto Σ-enumerable, deberemos encontrar un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ tal que $\text{Dom}\Psi^{1,0,\#}_{\mathcal{P}} = \omega$ y $\text{Im}\,\Psi^{1,0,\#}_{\mathcal{P}} = S$. Dicho en palabras, el programa $\mathcal{P}$ debera cumplir:

- siempre que lo corramos desde un estado de la forma $\|x\|$, con $x \in \omega$, debe detenerse y el contenido de la variable N1 bajo detencion debera ser un elemento de S
- para cada $s \in S$ debera haber un $x \in \omega$ tal que s es el valor de la variable N1 bajo detencion cuando corremos $\mathcal{P}$ desde $\|x\|$

A continuacion daremos una descripcion intuitiva del funcionamiento de $\mathcal{P}$ (pseudocodigo) para luego escribirlo correctamente usando macros. El programa $\mathcal{P}$ comenzara del estado $\|x\|$ y hara las siguientes tareas

Etapa 1: si $x = 0$ ir a Etapa 6, en caso contrario ir a Etapa 2.

Etapa 2: calcular $(x)_1$ y $(x)_2$ e ir a Etapa 3.

Etapa 3: si $\mathcal{P}_0$ termina desde $\|(x)_1\|$ en $(x)_2$ pasos ir a Etapa 4, en caso contrario ir a Etapa 6

Etapa 4: si el valor que queda en N1 luego de correr $\mathcal{P}_0$ una cantidad $(x)_2$ de pasos, partiendo de $\|(x)_1\|$, es distinto de 0, entonces ir a Etapa 5. En caso contrario ir a Etapa 6.

Etapa 5: asignar a N1 el valor $(x)_1$ y terminar

Etapa 6: asignar a N1 el valor 0 y terminar

Notese que la descripcion anterior no es ni mas ni menos que un procedimiento efectivo que enumera a S, y nuestra mision es simularlo dentro del lenguaje $\mathcal{S}^{\Sigma}$. Para esto usaremos varios macros. Ya que la funcion $f = \lambda x[(x)_1]$ es Σ-p.r., el Corolario 2 nos dice que hay un macro:

$$[\text{V2} \leftarrow f(\text{V1})]$$

el cual escribiremos de la siguiente manera mas intuitiva:

$$[\text{V2} \leftarrow (\text{V1})_1]$$

Similarmente hay un macro:

$$[\text{V2} \leftarrow (\text{V1})_2]$$

Tambien, ya que el predicado $P = \lambda x[x = 0]$ es Σ-recursivo, hay un macro:

$$[\text{IF } P(\text{V1}) \text{ GOTO A1}]$$

el cual escribiremos de la siguiente manera:

$$[\text{IF V1} = 0 \text{ GOTO A1}]$$

Definamos

$$H = \lambda tx \left[Halt^{1,0}(t, x, \mathcal{P}_0)\right]$$

Notar que $D_H = \omega^2$ y que H es Σ-mixta. Ademas sabemos que la funcion $Halt^{1,0}$ es $(\Sigma \cup \Sigma_p)$-p.r. por lo cual resulta facilmente que H es $(\Sigma \cup \Sigma_p)$-p.r..

Por la Proposicion de Independencia del Alfabeto tenemos que H es Σ-p.r.. O sea que el Corolario 2 nos dice que hay un macro:

$$[\text{IF } H(\text{V1}, \text{V2}) \text{ GOTO A1}]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\text{IF } Halt^{1,0}(\text{V1}, \text{V2}, \mathcal{P}_0) \text{ GOTO A1}\right]$$

Sea

$$g = \lambda tx \left[E^{1,0}_{\#1}(t, x, \mathcal{P}_0)\right]$$

Ya que g es Σ-recursiva (por que?), hay un macro:

$$[\text{V3} \leftarrow g(\text{V1}, \text{V2})]$$

Para hacer mas intuitivo el uso de este macro lo escribiremos de la siguiente manera

$$\left[\text{V3} \leftarrow E^{1,0}_{\#1}(\text{V1}, \text{V2}, \mathcal{P}_0)\right]$$

Ahora si podemos dar nuestro progama $\mathcal{P}$ que enumera a S:

	IF N1 $\neq 0$ GOTO L1
	GOTO L2
L1	[N3 $\leftarrow$ (N1)$_1$]
	[N4 $\leftarrow$ (N1)$_2$]
	$\left[\text{IF } Halt^{1,0}(\text{N4}, \text{N3}, \mathcal{P}_0) \text{ GOTO L3}\right]$
	GOTO L2
L3	$\left[\text{N5} \leftarrow E^{1,0}_{\#1}(\text{N4}, \text{N3}, \mathcal{P}_0)\right]$
	[IF N5 $= 0$ GOTO L2]
	N1 $\leftarrow$ N3
	GOTO L4
L2	N1 $\leftarrow 0$
L4	SKIP

Ejercicio 6: Sea $\Sigma = \{\#, \$\}$ y sea $f : D_f \subseteq \Sigma^* \rightarrow \omega$ una funcion Σ-computable tal que $f(\varepsilon) = 1$. Sea $L = \{\alpha \in D_f : f(\alpha) = 1\}$. De un programa $\mathcal{Q} \in \text{Pro}^{\Sigma}$ tal que $\text{Dom}(\Psi^{1,0,*}_{\mathcal{Q}}) = \omega$ y $\text{Im}(\Psi^{1,0,*}_{\mathcal{Q}}) = L$ (explicado en video en granlogico.com).

Ejercicio 7: Sea $\Sigma = \{\#, \$\}$ y sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$. Pruebe que

$$S = \{(x, \alpha) : \Psi^{1,0,\#}_{\mathcal{P}_0}(x) = \Psi^{0,1,\#}_{\mathcal{P}_0}(\alpha)\}$$

es Σ-enumerable.

Ejercicio 8: Si $S \subseteq \omega$ y $f : S \to \omega$ es Σ-computable, entonces el conjunto

$$\{x \in S : x \text{ es par } x/2 \in S \text{ y } f(x) = f(x/2)\}$$

es Σ-enumerable

Ejercicio 9: Sea $\Sigma = \{\#, \$\}$ y sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$. Pruebe que $\{(x, \alpha, \beta) \in \omega \times \Sigma^* \times \Sigma^* : \mathcal{P}_0$ termina partiendo de $\|x, \alpha, \beta\|\}$ es Σ-enumerable.

Ejercicio 10: Sea $\Sigma = \{@, !\}$ y sea $\mathcal{P}_0 \in \text{Pro}^{\Sigma}$. Sea

$$L = \{\alpha \in \Sigma^* : (\exists x \in \mathbf{N})\ \Psi_{\mathcal{P}_0}^{1,1,\#}(x^2, \alpha) = \Psi_{\mathcal{P}_0}^{0,2,\#}(\alpha, \alpha)\}$$

De un programa $\mathcal{P}$ tal que $\text{Dom}(\Psi_{\mathcal{P}}^{1,0,*}) = \omega$ y $\text{Im}(\Psi_{\mathcal{P}}^{1,0,*}) = L$.

Enumeracion de conjuntos de programas Ya que los programas de $\mathcal{S}^{\Sigma}$ son palabras del alfabeto $\Sigma \cup \Sigma_p$, nos podemos preguntar cuando un conjunto L de programas es $(\Sigma \cup \Sigma_p)$-enumerable. Daremos un ejemplo. Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \Psi_{\mathcal{P}}^{1,0,\#}(10) = 10\}$$

Veremos que L es $(\Sigma \cup \Sigma_p)$-enumerable, dando un programa $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ que enumere a L, es decir tal que $\text{Dom}(\Psi_{\mathcal{Q}}^{1,0,*}) = \omega$ y $\text{Im}(\Psi_{\mathcal{Q}}^{1,0,*}) = L$. Cabe destacar que aqui hay en juego dos versiones de nuestro lenguaje imperativo, es decir enumeraremos un conjunto de programas de $\mathcal{S}^{\Sigma}$ usando un programa de $\mathcal{S}^{\Sigma \cup \Sigma_p}$. Sea $\leq$ un orden total sobre el conjunto $\Sigma \cup \Sigma_p$.

A continuacion daremos una descripcion intuitiva del funcionamiento de $\mathcal{Q}$ (pseudocodigo) para luego escribirlo correctamente usando macros. Notese que SKIP $\in L$. El programa $\mathcal{Q}$ comenzara del estado $\|x\|$ y hara las siguientes tareas

Etapa 1: si $x = 0$ ir a Etapa 6, en caso contrario ir a Etapa 2.

Etapa 2: calcular $(x)_1$, $(x)_2$ y $*^{\leq}((x)_1)$ e ir a Etapa 3.

Etapa 3: si $*^{\leq}((x)_1) \in \text{Pro}^{\Sigma}$ y termina partiendo desde $\|10\|$ en $(x)_2$ pasos ir a Etapa 4, en caso contrario ir a Etapa 6

Etapa 4: si el valor que queda en N1 luego de correr $*^{\leq}((x)_1)$ una cantidad $(x)_2$ de pasos, partiendo de $\|10\|$ es igual a 10, entonces ir a Etapa 5. En caso contrario ir a Etapa 6.

Etapa 5: asignar a P1 la palabra $*^{\leq}((x)_1)$ y terminar

Etapa 6: asignar a P1 la palabra SKIP y terminar

Notese que la descripcion anterior no es ni mas ni menos que un procedimiento efectivo que enumera a L, y nuestra mision es simularlo dentro del lenguaje $\mathcal{S}^{\Sigma\cup\Sigma_p}$. Para esto usaremos varios macros. Es importante notar que los macros que usaremos corresponden al lenguaje $\mathcal{S}^{\Sigma\cup\Sigma_p}$ ya que los usaremos en $\mathcal{Q}$ el cual sera un programa de $\mathcal{S}^{\Sigma\cup\Sigma_p}$.

Ya que las funciones $\lambda x[(x)_1]$ y $\lambda x[(x)_2]$ son $(\Sigma\cup\Sigma_p)$-recursivas el Corolario 2 nos dice que hay macros asociados a estas funciones los cuales escribiremos de la siguiente manera mas intuitiva:

$$\begin{aligned}[\text{V2} &\leftarrow (\text{V1})_1]\\ [\text{V2} &\leftarrow (\text{V1})_2]\end{aligned}$$

Ya que el predicado $P = \lambda x[x = 10]$ es $(\Sigma\cup\Sigma_p)$-recursivo tenemos su macro asociado el cual escribiremos de la siguiente manera:

$$[\text{IF V1} = 10 \text{ GOTO A1}]$$

Por un lema anterior sabemos que Pro^{Σ} es un conjunto $(\Sigma\cup\Sigma_p)$-p.r.´por lo cual $\chi_{\text{Pro}^{\Sigma}}^{(\Sigma\cup\Sigma_p)^*}$ es $(\Sigma\cup\Sigma_p)$-p.r., por lo cual hay un macro

$$\left[\text{IF } \chi_{\text{Pro}^{\Sigma}}^{(\Sigma\cup\Sigma_p)^*}(\text{W1}) \text{ GOTO A1}\right]$$

el cual escribiremos de la siguiente manera

$$\left[\text{IF W1} \in \text{Pro}^{\Sigma} \text{ GOTO A1}\right]$$

Ya que el predicado $Halt^{1,0}$ es $(\Sigma\cup\Sigma_p)$-recursivo tenemos un macro asociado a el, el cual escribiremos de la siguiente forma

$$\left[\text{IF } Halt^{1,0}(\text{V1}, \text{V2}, \text{W1}) \text{ GOTO A1}\right]$$

Ya que $E_{\#1}^{1,0}$ es $(\Sigma\cup\Sigma_p)$-recursivo tenemos un macro asociado a ella, el cual escribiremos de la siguiente forma

$$\left[\text{V3} \leftarrow E_{\#1}^{1,0}(\text{V1}, \text{V2}, \text{W1})\right]$$

Tambien usaremos macros

$$\begin{aligned}[\text{V1} &\leftarrow 10]\\ [\text{W1} &\leftarrow \text{SKIP}]\end{aligned}$$

(dejamos al lector hacerlos a mano o tambien se puede justificar su existencia via la proposicion de existencia de macros aplicada a las funciones $C_{10}^{0,0}$ y $C_{\text{SKIP}}^{0,0}$).

Ahora si podemos hacer el programa $\mathcal{Q}$ que enumera a L:

$$
\begin{array}{ll}
 & \text{IF N1} \neq 0 \text{ GOTO L1} \\
 & \text{GOTO L2} \\
\text{L1} & [\text{N2} \leftarrow (\text{N1})_1] \\
 & [\text{N3} \leftarrow (\text{N1})_2] \\
 & [\text{P1} \leftarrow *^{\leq}(\text{N2})] \\
 & \left[\text{IF P1} \in \text{Pro}^{\Sigma} \text{ GOTO L3}\right] \\
 & \text{GOTO L2} \\
\text{L3} & [\text{N4} \leftarrow 10] \\
 & \left[\text{IF } Halt^{1,0}(\text{N3}, \text{N4}, \text{P1}) \text{ GOTO L4}\right] \\
 & \text{GOTO L2} \\
\text{L4} & \left[\text{N5} \leftarrow E^{1,0}_{\#1}(\text{N3}, \text{N4}, \text{P1})\right] \\
 & [\text{IF N5} = 10 \text{ GOTO L4}] \\
\text{L2} & [\text{P1} \leftarrow \text{SKIP}] \\
\text{L4} & \text{SKIP}
\end{array}
$$

Ejercicio 11: Sea $\Sigma = \{@, !\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists \alpha \text{ tal que } \Psi^{1,1,*}_{\mathcal{P}}(11, \alpha) = \alpha!@\}$$

(a) Dar un programa $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ tal que $\text{Dom}(\Psi^{0,1,\#}_{\mathcal{Q}}) = L$

(b) Dar un programa $\mathcal{Q}' \in \text{Pro}^{\Sigma \cup \Sigma_p}$ tal que $\text{Dom}(\Psi^{1,0,*}_{\mathcal{Q}'}) = \omega$ y $\text{Im}(\Psi^{1,0,*}_{\mathcal{Q}'}) = L$

Ejercicio 12: (Explicado en video en granlogico.com.) Sea $\Sigma = \{\#, \$\}$ y sea

$$L = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists n, \alpha \text{ tales que } \Psi^{2,1,*}_{\mathcal{P}}(|\mathcal{P}|, n, \alpha) = \$\}$$

(a) Dar un programa $\mathcal{Q} \in \text{Pro}^{\Sigma \cup \Sigma_p}$ tal que $\text{Dom}(\Psi^{0,1,\#}_{\mathcal{Q}}) = L$

(b) Dar un programa $\mathcal{Q}' \in \text{Pro}^{\Sigma \cup \Sigma_p}$ tal que $\text{Dom}(\Psi^{1,0,*}_{\mathcal{Q}'}) = \omega$ y $\text{Im}(\Psi^{1,0,*}_{\mathcal{Q}'}) = L$

Cuando $\Sigma \supseteq \Sigma_p$ podemos correr un programa $\mathcal{P} \in \text{Pro}^{\Sigma}$ partiendo de un estado que asigne a sus variables alfabeticas programas (ya que los programas son meras palabras de Σ^*). En particular podriamos correr un programa $\mathcal{P}$ desde el estado $\|\mathcal{P}\|$. Llamaremos A al conjunto formado por aquellos programas $\mathcal{P}$ tales que $\mathcal{P}$ se detiene partiendo del estado $\|\mathcal{P}\|$. Es decir

$$A = \{\mathcal{P} \in \text{Pro}^{\Sigma} : \exists t \in \omega \text{ tal que } Halt^{0,1}(t, \mathcal{P}, \mathcal{P}) = 1\}$$

Por ejemplo SKIP $\in A$. Dicho rapida y sugestivamente A es el conjunto formado por aquellos programas que se detienen partiendo de si mismos.

Ejercicio 13: Supongamos que $\Sigma \supseteq \Sigma_p$. De un programa de $\mathcal{S}^\Sigma$ que enumere a A.

Ejercicio 14: Supongamos que $\Sigma \supseteq \Sigma_p$. Sea

$$L = \{\mathcal{P} \in \text{Pro}^\Sigma : \Psi_{\mathcal{P}}^{2,1,*}(1,1,\mathcal{P}) = \mathcal{P}\mathcal{P}\}$$

(a) Encuentre un elemento concreto de L

(b) De un programa de $\mathcal{S}^\Sigma$ que enumere a L

Dos batallas mas

Aceptaremos sin prueba el siguiente teorema.

Theorem 12 (Godel vence a Turing) *Supongamos $f : S \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-Turing computable. Entonces f es Σ-recursiva.*

Ejercicio 15: (S) Haga un esquema a nivel de ideas (sin demostraciones y sin demaciada precision) de la prueba del teorema anterior (la prueba completa esta en el apunte)

Aceptaremos sin prueba el siguiente teorema.

Theorem 13 (Turing vence a Neumann) *Si $f : D_f \subseteq \omega^n \times \Sigma^{*m} \to O$ es Σ-computable, entonces f es Σ-Turing computable.*

Ejercicio 16: (S) Haga un esquema a nivel de ideas (sin demostraciones y sin demaciada precision) de la prueba del teorema anterior (la prueba completa esta en el apunte y en granlogico.com hay un video con la prueba del teorema)

La tesis de Church

En virtud de los teoremas ya probados tenemos el siguiente teorema que asegura que los tres paradigmas son equivalentes.

Theorem 14 *Sea Σ un alfabeto finito. Dada una funcion f, las siguientes son equivalentes:*

(1) f es Σ-Turing computable

(2) f es Σ-recursiva

(3) f es Σ-computable.

Tambien los tres paradigmas son equivalentes con respecto a los dos tipos de conjuntos estudiados, es decir:

Theorem 15 *Sea Σ un alfabeto finito y sea $S \subseteq \omega^n \times \Sigma^{*m}$. Las siguientes son equivalentes:*

(1) S es Σ-Turing enumerable

(2) S es Σ-recursivamente enumerable

(3) S es Σ-enumerable

Theorem 16 *Sea Σ un alfabeto finito y sea $S \subseteq \omega^n \times \Sigma^{*m}$. Las siguientes son equivalentes:*

(1) S es Σ-Turing computable

(2) S es Σ-recursivo

(3) S es Σ-computable

Ejercicio 17: Pruebe los dos teoremas anteriores

Otro modelo matematico de computabilidad efectiva es el llamado lambda calculus, introducido por Church, el cual tambien resulta equivalente a los estudiados por nosotros. El hecho de que tan distintos paradigmas computacionales hayan resultado equivalentes hace pensar que en realidad los mismos han tenido exito en capturar la totalidad de las funciones Σ-efectivamente computables. Esta aceveracion es conocida como la

Tesis de Church: *Toda funcion Σ-efectivamente computable es Σ-recursiva.*

Si bien no se ha podido dar una prueba estrictamente matematica de la Tesis de Church, es un sentimiento comun de los investigadores del area que la misma es verdadera.